人工知能学会研究会資料
SIG-FAI-9703-17(3/27)

記事連番0017

強化学習によるオセロの戦略の獲得

Strategy acqusition for the "Othello" game based on the reinforcement learning

吉岡 琢 Taku Yoshioka
石井 信 Shin Ishii
伊藤 実 Minoru Ito
奈良先端科学技術大学院大学
Nara Institute of Science and Technology

Abstract

This research report discusses automatic strategy acqusition for the othello game based on the reinforcement learning. In our approach, two computer players initially know only the game rules, but become fairly strong after playing many games with each other. In each game, the players refine the evaluation of the game state, which is achieved in a reinforcement learning manner. Since the state space is too large, we employ a RBF(Radial Basis Functions) model to approximate the evaluation function. After several thousand games, the players becomes strong enough to beat a player employing a heuristic strategy.

1 はじめに

強化学習は試行錯誤の繰り返しによって、最適な戦略を獲得する機械学習の手法である [7]。この手法では、ある戦略をとったとき、その結果に応じた報酬を与えることによって、最適な戦略、すなわち得られる報酬を最大にする戦略を獲得する。本研究では強化学習を6×6のオセロに応用して、ルール以外の知識を持たないプレイヤー同士が繰り返し対戦することにより、戦略を自動的に獲得できることを示す。

強化学習のゲームへの応用例として、バックギャモン [1] [2][3]、チェス [4]、碁 [9] などがある。特にバックギャモンは、世界チャンピオンに匹敵する戦略を獲得することに成功している。バックギャモンをプレイする、Tesauro の TD-Gammon は、TD(λ) [10] と呼ばれる強化学習のアルゴリズムを用いて、最適な戦略を獲得する。戦略を獲得した TD-Gammon は、各ステップで盤面の状態をニューラルネット（多層パーセプトロン）へ入力して、その出力（評価関数）に基づいて打つ手を決める。解の探索は行なわれず、先読みのような知識は TD(λ) によって学習された評価関数に含まれると考えることができる。

強化学習をゲームに応用する時の問題点として、評価関数の表現方法がある。表現方法は大きく2つに分けられる [1]。1つはテーブルによる表現、もう一つは関数近似による表現である。バックギャモンやオセロのようなゲームは状態空間が非常に大きいため、テーブルによる表現は現在のコンピュータでは現実的に困難で

ある。この理由から、ニューラルネットのような関数近似器を用いた評価関数の表現が必要である。

オセロの場合、さらに二つの問題が存在する。一つは、オセロがバックギャモンと異なり確率的要素がないことである。Tesauroは、バックギャモンが確率的要素を含むことが、TD-Gammonが成功するための要素であったことを指摘している[1]。確率的要素が果たす役割の一つは、状態空間を幅広く探索することである。オセロで同じ事を行なうためには、意図的に確率的な要素を入れる必要がある。2章でこの方法について述べる。

もう一つの問題は、オセロでは局面変化が激しく、盤面の状態から評価への関数は非常に起伏の大きなものとなることである。この点で先に述べたバックギャモン、チェス、碁などと異なる。こうした際に、ニューラルネットで十分に関数を近似できるか、あるいは未知の状態についても適当に外挿（汎化）できるかどうかは重要な課題である。

本研究の手法では、ルール以外の知識を持たない2つのプレイヤーを用意する。それらはRBF(Radial Basis Function)ネットワークを評価関数として用いる。学習前はネットワークは適当に初期化されていて、ランダムにプレイする。そのようなプレイヤー同士が対戦を行ない、対戦が終了した時点で結果に応じた報酬を得て、それに基づいてネットワークの学習が行なわれる。対戦を数千回以上繰り返すことによって、両方のプレイヤーは次第に正しい評価関数を学習し、それに基づいた強い戦略を獲得する。学習の評価は、学習を行なったプレイヤーをヒューリスティックな戦略を持ったプレイヤーと対戦させることによって行なう。

2 戦略と学習

本研究では、コンピュータプレイヤーは評価関数を用いて、1手先を読んでmin-max戦略をとることによって、次の手を決定する。もし、ずっと先にわたる局面の変化まで考慮された評価を出力する評価関数を獲得することができれば、探索をほとんど行なわずに手を決めることができる。一方で、評価関数はコンピュータプレイヤー同士が対戦を繰り返す過程で獲得される。すなわち、人間がオセロの評価に関する知識を与える必要はない。

この章ではコンピュータプレイヤーの戦略、評価関数の表現およびその学習方法、そして学習のプロセスについて説明する。

2.1 戦略

説明のために、いくつかの記号を定義する。黒（先手）と白のプレイヤーを考える。各プレイヤーの手数tは1から始まり、両方のプレイヤーが手を打つごとに1ずつ増える。黒と白が打つ手を$a_t^\bullet$、a_t°のように表す。以下、プレイヤーが黒の場合について説明するが、白の場合でも同様である。手数t、黒の番における盤面の状態ベクトルを$s_t^\bullet$とする。状態ベクトル$s_t^\bullet$に対する黒の評価関数を$\hat{V}^\bullet(s_t^\bullet)$とする。各プレイヤーはそれぞれ別々に評価関数を持つ。盤面の状態が$s_t^\bullet$のときに手$a_t^\bullet$を打った後の状態を$s_t^\circ = S(s_t^\bullet, a_t^\bullet)$で表す。ここでは、黒が手を打ったことにより、次の番面は相手の順番(○)になっている。

ある時点で、盤面の状態が$s_t^\bullet$であるとする。このとき、次式を満足するような手$a_t^\bullet$を次の手とする。

$$a_t^\bullet = \mathrm{argmax}_{a_t^{\bullet\prime}}\{P_{\min}^\bullet(s_t^{\circ\prime})\} \quad (1)$$

$$s_t^{\circ\prime} = S(s_t^\bullet, a_t^{\bullet\prime}) \quad (2)$$

$$P_{\min}^\bullet(s_t^{\circ\prime}) = \min_{a_t^{\circ\prime}}\{\hat{V}^\bullet(s_{t+1}^{\bullet\prime})\} \quad (3)$$

$$s^{\bullet}_{t+1}{}' = S(s^{\circ}_{t}{}', a^{\circ}_{t}{}') \quad (4)$$

これは min-max 戦略である。このとき、式 (3) より、白が黒の評価関数に基づいて最適な戦略（黒にとって最も悪い戦略）をとると仮定している。つまり、黒は白の評価関数を用いずに、自らの評価関数で代用した min-max 戦略により、次に打つ手を決定する。

2.2 評価関数の表現

次に評価関数の表現について考える。ここで、もし状態空間が小さければ、評価関数はテーブルによって表現できる。しかし、6 × 6 のオセロの場合、盤面が取り得る状態の数は 3^{36} 程度であり[1]、テーブルによる表現は現実的には困難である。そこで本手法では RBF ネットワークによって評価関数を近似的に表す。

盤上で白が置かれているマスを 1、を黒がおかれているマスを −1、何も置かれていないマスを 0 にそれぞれ対応させて、盤のマスの数の成分を持つベクトルとして、盤面の状態を表現する。評価関数は盤面の状態を入力として、将来ゲームが終了したときの黒の駒数と白の駒数の差を出力することを期待する。

盤面の状態ベクトルが s のとき、RBF ネットワーク [8] によって評価関数の値 $\hat{V}(s)$ は次のように計算される。

$$\hat{V}(s) = \sum_{i=1}^{N} w_i g_i(s) \quad (5)$$

$$g_i(s) = \frac{\phi_i(s)}{\sum_{j=1}^{N} \phi_j(s)} \quad (6)$$

$$\phi_i(s) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-|s-\mu_i|^2}{2\sigma^2}\right) \quad (7)$$

ここで N は RBF の数、w_i は各 RBF の重み、μ_i は各 RBF の状態空間上での位置、σ は μ_i の分散を表している。

[1] 対称性や、最初から置かれている 4 つの駒を考えるともっと少なくなるが、それでもテーブルによる表現は困難である。

2.3 RBF ネットワークの学習

学習は、RBF ネットワークのパラメータ w_i、および μ_i を、次の二乗誤差が最小になるように更新することによって行なう。

$$E = \sum_{t=1}^{t_{max}} \{V_t - \hat{V}(s_t)\}^2 = \sum_{t=1}^{t_{max}} E_t^2 \quad (8)$$

ここで、t_{max} はゲームが終了した手数、V_t は $\hat{V}(s_t)$ に対する教師信号であり、

$$V_t = \begin{cases} \text{黒の駒数} - \text{白の駒数} & (t = t_{max}) \\ \max_{a^{\circ}_t{}'}\{P_{min}(s^{\circ}_t{}')\} & (\text{それ以外}) \end{cases} \quad (9)$$

最急降下法を用いて Δw_i、$\Delta \mu_i$ を計算する。

$$\Delta w_i = -\eta_w \frac{\partial E}{\partial w_i} = -\eta_w \sum_{t=1}^{t_{max}} E_t g_i(s_t) = -\eta_w \sum_{t=1}^{t_{max}} \Delta w_{it} \quad (10)$$

$$\Delta \mu_i = -\eta_\mu \frac{\partial E}{\partial \mu_i} = -\eta_\mu \sum_{t=1}^{t_{max}} E_t g_i(s_t) \times (s_t - \mu_i)(w_i - \hat{V}_t) = -\eta_\mu \sum_{t=1}^{t_{max}} \Delta \mu_{it} \quad (11)$$

ここで、η_w、$\eta_\mu > 0$は学習係数である。ゲームが1回終了するごとに、以上のようにパラメータを更新する。また、Δw_{it}、$\Delta \mu_{it}$は、手を打つ時点で（対戦が終了するのを待たずに）計算できる。

以上の説明より、ゲームの途中の盤面に対しては、min-maxによって得られる1手先の評価値の中で最大のものを教師信号とし（図1）、ゲームが終了した時点の盤面に対しては、盤上の駒数の差を教師信号として与える。そして、ネットワークが教師信号に近い値を出力するように、最急降下法による学習を行なう。この方法がうまくいく直観的な理由は次のようになる。まず、ゲームが終了した状態については、盤面の評価を正しく学習できる。そして、全ての終了状態に対する評価を正しく学習できたと仮定すると、それを用いて一手前の評価を正しく学習することができる。このように、ゲームの終了状態から逆向きに学習が進むことが期待できる。

2.4 学習のプロセス

この章の冒頭で述べたように、学習はコンピュータプレイヤー同士の対戦を繰り返すことによって行なわれる。各プレイヤーはそれぞれ評価関数を表現するためにRBFネットワークを持つ。対戦の中で、RBFネットワークのパラメータを更新することにより学習が進行する。学習のプロセスをまとめると次のようになる。

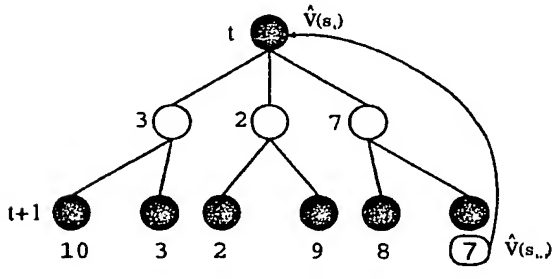

図 1: 評価関数の学習

1. 各プレイヤーのRBFネットワークのパラメータを初期化する。
2. ゲームを開始する。
3. ゲームが終了するまで、手を打つと同時にΔw_{it}、$\Delta \mu_{it}$を計算する。
4. ゲームが終了した時点で、RBFネットワークのパラメータを式(10)、(11)で更新する。
5. (2)へ戻る。

以上で説明した手法では、確率的な要素が一切入っていない。このため、同じ盤面ばかりを学習することになる。これを避けるために、プレイヤーが手を決めるときに、評価に乱数を加える。一方、学習が進行するにつれて、乱数が評価関数を過小評価して、評価の高い手を選択することを妨げる。そこで、学習が進行するにつれて乱数の大きさを下げることによって、評価の高い手の先を重点的に学習させるようにする。式(1)を次のように変更する。

$$a_t^\bullet = \mathrm{argmax}_{a_t^\circ}\{P_{\min}^\bullet(s_t^{\circ\prime}) + \mathrm{rnd}(T)\} \quad (12)$$

ここでTはゲームの回数（手数ではない）、$\mathrm{rnd}(T)$として、ここでは平均0、分散が$f(T)$の正規乱数を用いる。$f(T)$は、Tについて線形に減少する関数である。

3 実験結果

以上の説明に基づいて、オセロの学習を行なうプログラムを作成した。図2はその実行画面である。これは人間と学習後のプレイヤーが対戦している所である。

このプログラムを用いて、次のような実験を行なった。まず、プレイヤー同士を対戦させて学習を行なう。RBFのユニット数は300とし、学習前に、w_iは0、μ_iの各成分は-1から1までの一様乱数として初期化する。対戦は10000回行ない、100回の対戦ごとに、学習データを黒と白のプレイヤーで分けて保存する。これら（$RL^{\bullet}$、RL°で表す）を、次のようなルールベースの戦略を持つプレイヤー（$RB^{\bullet}$、RB°で表す）と対戦させて、その勝率を調べる。

- 最後の3手では、min-maxによる全解探索により最善手を選ぶ
- それ以外では、次の値を最大にするような手を選ぶ

$$\frac{\text{手を打った後の自分と相手の駒の数の差}}{\text{手を打った後の駒の数}}$$

+(手が盤面の4つの角のいずれかならば1.0)

+(平均0、分散0.1の正規乱数)

図3、4は、それぞれ$RL^{\bullet}$とRB°、$RB^{\bullet}$とRL°の、対戦時の勝率の変化を表す。横軸が$RL^{\bullet}$（RL°）の学習回数、縦軸が$RL^{\bullet}$（RL°）のRB°（$RB^{\bullet}$）に対する勝率を表す。未学習の状態では一割程度の勝率だったのが、学習を行なうことによって最大で九割以上勝てるようになることが分かる。勝率は学習回数につれて単純に増加しているわけではなく、勝率が大きく下がっている部分もある。これはRBFの学習能力や学習時のランダムネス等に起因すると考えられる。

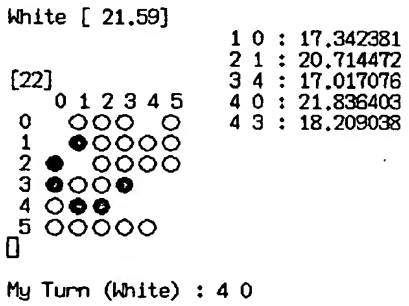

図2: プログラムの実行画面

4 考察

自己対戦学習、すなわちコンピュータ同士を対戦させながら強化学習させることにより、ルール以外の知識を持たない状態から、ある程度強い戦略が獲得できた。この要因の一つとして、対戦回数を比較的多くできたということが考えられる。これは、オセロが比較的単純なゲームであるからできることである。例えばチェスでは、ゲームの複雑さのためそれほど対戦回数を多くできず、自己対戦学習では良い結果を得ることができないことが報告されている[4]。一方、自己対戦学習によって成功したバックギャモンでは150万回もの対戦を行なっている[2]。

また、我々は以前に多層パーセプトロン（ユニット数250）を用いてオセロの自動学習を行なった[6]が、本研究ではそれよりも少ない対戦回数で、同程度に強い戦略を獲得している。この原因の一つとして、多層パーセプトロンとRBFネットワークの関数近似の方法の違いが考えられる。多層パーセプトロンは、一つのデータを学習する時に全てのユニットが関係するた

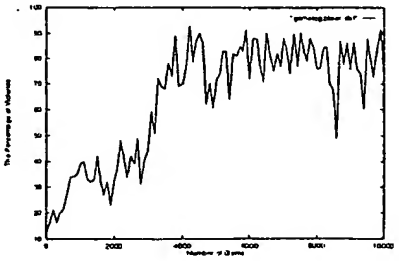

図3: $RL^{\bullet}$対RB°の勝率の変化

め、学習データ以外の状態の値を変えてしまう。多層パーセプトロンのような大域モデルで強化学習を行なうと、学習が発散するという報告もある [5]。一方、RBF は状態空間上で近い位置にあるユニットが主に学習に関係するため、多層パーセプトロンに比べて、他の状態に与える影響が少ない。そして、本研究ではオンライン的に学習を行なっているため、局所モデルである RBF が適していると考えられる。

オセロの場合、状態空間は大きいが、ゲームの中で全ての盤面が均等な割合で出現するわけではない。RBF ネットワークは、構成要素である RBF が状態空間を移動することによって、頻繁に出現する状態の近傍を正確に近似しようとする。つまり、RBF は一種の状態空間の特徴検出を行なっている。そのため、わずか 300 個のユニットで 3^{36} の状態空間に対する関数を表現することができると考えられる。

5 まとめ

本研究では、強化学習によってオセロの戦略を獲得する手法を提案した。評価関数の表現方法として RBF ネットワークを用いた。知識を持たないプレイヤー同士が対戦を繰り返すことによって、ある程度強い戦略を獲得できることを、ルールベースのプレイヤーとの対戦によって実験的に確かめた。

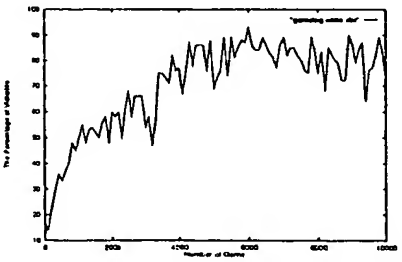

図 4: *RB*•対 *RL*°の勝率の変化

参考文献

[1] Gerald Tesauro, *Machine Learning* **8**, pp. 257-277, 1992.

[2] Gerald Tesauro, *Neural Computation*, **6**, pp. 215-219, 1994.

[3] Gerald Tesauro, *Communications of the ACM*, **38** pp. 58-68, 1995.

[4] Jonathan Baxter, Andrew Tridgell, Lex Weaver, *Technical Report*, Canberra: Australian National University, 1997

[5] Justin A. Boyan, Andrew W. Moore, *Advances in Neural Information Processing Systems* **7**, pp. 369-376, The MIT Press, 1995

[6] 林 則昌, 石井 信, *ATR Technical Report*, **TR-H-159** Kyoto: ATR, 1995

[7] Leslie P. Kaelbling, Michael L. Littman, Andrew W. Moore, *Journal of Artifical Intelligence Research* **4**, pp. 237-285, 1996

[8] John Moody, Christian J. Darken, *Neural Computation* **1**, pp. 281-294, 1989

[9] Nicol N. Schraudolph, Peter Dayan, Terrence J. Sejnowski, *Advances in Neural Information Processing* **6**, pp. 817-824, Morgan Kaufmann, 1994

[10] Richard S. Sutton, *Machine Learning* **3**, pp. 9-44, Boston: Kluwer, 1988